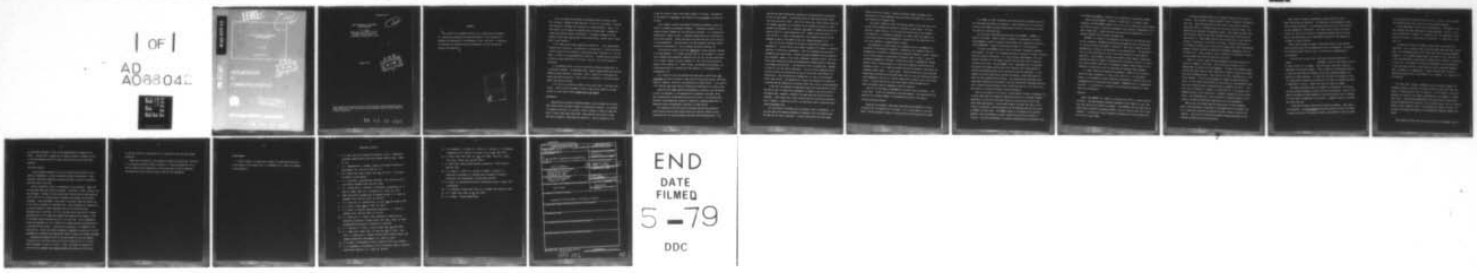
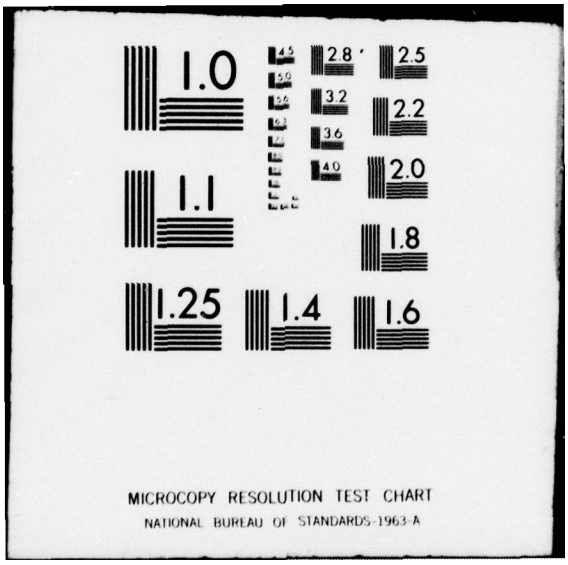
AD-A066 042 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 9/4
THE INFLUENCE OF ALGORITHMS AND HEURISTICS.(U)
JAN 79 J F TRAUB
UNCLASSIFIED CMU-CS-79-102 N00014-76-C-0370

| OF |
AD
A066042



END
DATE FILMED
5-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL II

12 NRL

THE INFLUENCE OF ALGORITHMS and HEURISTICS

J. F. Traub

January 1979

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

CMU-CS-79-102

(12)

THE INFLUENCE OF ALGORITHMS AND HEURISTICS

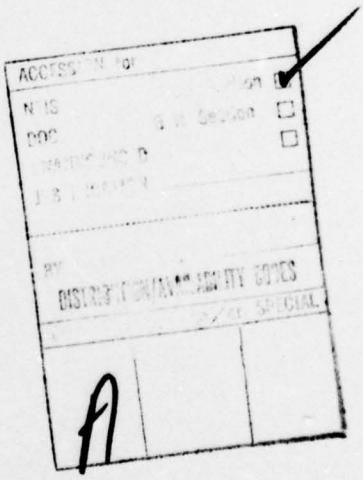
J. F. Traub
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

January 1979

This research was supported in part by the National Science Foundation under Grant MCS75-222-55 and the Office of Naval Research under Contract N00014-76-C-0370, NR 044-422.

ABSTRACT

This article is an expanded version of an invited address presented at a symposium celebrating the Tenth Anniversary of IRIA (Institute de Recherche d'Informatique et d'Automatique), Paris, June 1978. It discusses the influence of procedural ideas on mathematics, science and applied science, and education.



I will talk about the influence of procedural ideas on various fields. This influence, although already important, is still in its infancy. I believe that during our lifetimes this influence will become pervasive. Here, I will limit myself to the impact of procedural ideas on three areas: mathematics, science and applied science, and education. Although my examples will be primarily drawn from research done in the United States, similar work is being done in many countries.

I will often use the words algorithm and heuristic. I will define these concepts only intuitively. An algorithm is a recipe and if I follow the recipe a certain result will be achieved. On the other hand, a heuristic is a rule of thumb. A result is not guaranteed for a heuristic. Instead results are obtained which are good enough most of the time. See (1), (2) for more extensive discussion.

It is sometimes useful to view an algorithm as being an extreme point of a continuum of methods. As knowledge about a domain becomes formalized, heuristic methods are often replaced by algorithms. When I choose not to distinguish between algorithm and heuristic I shall refer to a procedure, a term which encompasses the entire continuum.

I will not try to survey the impact of procedural ideas; that would take a book. I want to give the reader a taste of some current research focusing primarily on work much of whose <u>impact lies in the future</u>.

Mathematics

Logicians have long been interested whether certain problems are solvable. More recently there has been interest in answering the question: If a problem is solvable, how hard is it to solve? Given a problem, is there an algorithm which is faster than known algorithms? These questions have led to a major new area of research: algorithms and complexity. See for example (3), (4)

to get the flavor of some of the recent research in the area. The addition of the question of how hard to the question of is it solvable has been very fruitful.

As an example, consider the problem of determining whether a given sentence is a theorem in Presburger arithmetic. (Presburger arithmetic is the system of natural numbers with the operation of addition. It is therefore an exceptionally "simple" system.) Presburger showed there is an algorithm for solving this problem. Fischer and Rabin (5) proved that the cost of solution is super-exponential. The implications for both automatic and human theorem-proving are controversial; see for example the panel discussion (6).

The difficulty (usually called the complexity) of many important problems is an open question. It is widely believed that all the problems in a certain set of problems have exponential complexity but this has not been established. It is of great interest to resolve whether "P = NP", which would provide an answer to this question (7). Rivest, Shamir, and Adleman have proposed a surprising application of a problem with high complexity to cryptography and cryptoanalysis (8).

As a contrast to the hard problems discussed above, astonishingly fast algorithms have been discovered in many areas including graph theory, operations research, geometry, statistics, and manipulation of power series. To be specific I will mention some recent results concerning the manipulation of power series.

Among the most common operations performed by scientists, applied mathematicians, and engineers is the manipulation of polynomials and power series. Basic manipulations include multiplication and division. More advanced manipulations include powering, reversion, composition, repeated composition, or computing certain transcendental functions of a power series.

The invention of the Fast Fourier Transform implied that polynomial multiplication could be done faster than by the classical method. Do fast algorithms exist for other polynomial and power series manipulations? This

question has been affirmatively answered for the manipulations listed above as well as many others. The new algorithms are often astonishingly faster than those previously known. Some of the results are highly counter-intuitive. For example, any power of a polynomial can be computed as quickly as simply squaring the polynomial (9). Moreover, any number of compositions can be computed as fast as a single composition (10). Another surprising result is that the first N terms of the expansion of any algebraic function can be computed as fast as the product of two Nth degree polynomials (11).

What will be the influence of the work on algorithms and complexity on mathematics? I believe we will see a major resurgence of interest in answering questions in constructive mathematics. Until the nineteenth century much of mathematics was constructive. Consider the great mathematicians who worked on such problems: Newton, Euler, Gauss, Chebyshev, Lagrange, Fourier, and so on. Then a major change took place and mathematicians became primarily interested in existence and structure. I believe we will see a paradigm shift. Mathematicians will ask new kinds of questions -- not just whether something exists but how it may be constructed and the cost of the optimal algorithm for its construction. The new constructive mathematics will have a very different flavor from the old. The questions will concern the complexity of classes of algorithms and the complexity of problems rather than of only a single algorithm.

I must add I've expected for some twenty years that there would be a great new wave of work in constructive mathematics, although it has not occurred to the degree that I expected. Recently I've seen some signs that this may be changing. I must stress that I don't believe that only constructive questions are of interest -- merely that we will see additional kinds of questions being asked by mathematicians.

I'll now discuss a different use of procedural ideas in mathematics. As you know one of the greatest mathematical problems of the last hundred years has been the four color conjecture. A proof eluded some of the best mathe-

matical minds for a century. Recently Professors Appel and Haken of the University of Illinois announced they had proven the theorem (12) using well over 1000 hours of computer time.

They used a computer to deal with the many special cases that occur after the four color problem was reduced to a problem in graph theory. This was certainly a major feat but it is not the only possible use of the computer in this connection. An interesting question is to what extent the computer can in the future assist in such mathematical discoveries as the reduction of the four color problem to a graph theoretic problem.

More generally, to what extent can a computer serve as a mathematician's assistant in the proving of theorems? Furthermore, can a computer serve as a mathematician's assistant in conjecturing interesting theorems to prove? Research on the latter question is being done by Professor Lenat (13) who is trying to understand mathematical discovery. Unlike earlier work in mathematical theorem proving, Lenat's program chooses for itself which concepts to define and which theorems to prove. It is creating concepts and proving theorems in set theory and elementary number theory. To quote Lenat: "There has been very little published thought about discovery from the algorithmic point of view; even Polya and Poincare treat mathematical ability as a sacred, almost mystic quantity, tied to the unconscious. It may be possible to learn from theorem finding programs how to tackle the general task of automating scientific research."

What will be the effect on mathematics of Lenat's work on discovery -- that is not clear. It will be interesting to see what happens. This research has the potential for profound changes in how mathematicians work in the future.

Science and Applied Science

Computers and procedural ideas have transformed large segments of science. In keeping with the objective of this paper, I will confine myself to one area where I believe procedural ideas will have pervasive and profound future impact.

In a number of areas, researchers have been working on programs which will be intelligent assistants for scientists. These assistants work in narrow but difficult task domains and help human experts do some of the taxing but essential parts of a particular job.

The first such intelligent assistant was called DENDRAL. DENDRAL is a chemist's assistant. Others include a scientist's and engineer's assistant called MACSYMA, a doctor's assistant called MYCIN, and a geologist's assistant called PROSPECTOR. I want to discuss each of these in a bit more detail.

I'll begin with DENDRAL which is due to Professors Feigenbaum, Lederberg, and Buchanan. Its task is to enumerate plausible structures for organic molecules given two kinds of information: [1] data from mass spectrometers, [2] user supplied constraints on the answer, derived from any other form of knowledge available to the user. How was DENDRAL built? Feigenbaum and his colleagues discovered that chemists who do this kind of analysis use a large amount of specialized knowledge without which the analysis would be impossible. So the computer scientists observed and studied the chemists, asking them questions about how they conducted their analyses. They organized the algorithmic knowledge about connectivity and valences, and also the heuristic knowledge about how such a scientist makes particular kinds of decisions when he is not really sure, when there is a variety of evidence, and much ambiguity.

What are the results of this work? As described by Professor Feigenbaum (14), in those areas where the program has been given specialist's knowledge, DENDRAL's performance is usually not only much faster but also much more accurate than expert human performance. To date some 25 papers have been published in major journals of chemistry, reporting results and the knowledge that had to be given to DENDRAL to obtain them. The DENDRAL system is in everyday use by Stanford chemists, their collaborators at other universities and also by chemists in industry. The British government is currently supporting work at Edinburgh aimed

at transferring DENDRAL to industrial user communities in the United Kingdom.

Before leaving DENDRAL, I mention a related effort in automatic theory formation, the META-DENDRAL project (15), which tries to bypass the time-consuming and difficult process of obtaining knowledge of mass spectral fragmentation rules from a human expert, instead extracting the regularities automatically (i.e. by computer program) from collected instrument data.

MACSYMA is a scientist's and engineer's assistant built by Professor Moses and his co-workers. MACSYMA does manipulative mathematics such as manipulation of rational functions and symbolic integration. Much time is spent by applied mathematicians, scientists and engineers on such manipulations. The question of how to do this efficiently has led to significant advances in algorithms and complexity. As one example, consider the problem of exact symbolic integration. Although the theory of symbolic integration was initiated by Liouville, it wasn't until quite recently that Risch (16) obtained a theory of when integrals can be computed in closed form as well as algorithms for obtaining these integrals.

I'll merely mention the remaining two examples. MYCIN (17) was started by Shortliffe (who, incidentally, is both a Ph.D. and a M.D.). It aids medical doctors in the diagnosis of blood infections and meningitis infections and in the recommendation of an antibiotic drug treatment. PROSPECTOR (18) is a geologist's assistant. It can serve as a consultant to aid exploration geologists in their search for ore deposits.

Education

Recall that DENDRAL is a chemist's assistant while MYCIN is a medical doctor's assistant. Yet these programs represent knowledge of their domains in such clear ways that they are also helpful to students. Thus DENDRAL rules have been used to teach organic structure elucidation; MYCIN's rules will soon be used (19) to teach diagnosis. I believe this view of teaching as procedure will prove valuable and will occur in many fields.

I'm going to describe some work of Professor Papert on the teaching of children. The following is based on material from P. McCorduck's forthcoming book (20). Papert believes that children learn by doing and by thinking about what they do. Although he does not see eye to eye with Piaget on all aspects he was profoundly influenced by Piaget. Both Piaget and Papert have made extensive inquiries into what children believe about learning and why they believe it. Among common theories that children hold is that learning consists of getting it, in a flash, all at once. Children who believe in this theory of learning lack or even resist a model which allows understanding gradually through a process of additions, refinements, debugging, and so on. As Papert says: "These children's ways of thinking is antithetical to learning any concept that cannot be acquired in one bite."

Papert is trying to teach children mathematics by having them learn to instruct a computer to do things. One of the ways he achieves this is through what his group calls Turtle Geometry. The name Turtle Geometry comes from the fact that there are small mechanisms at Papert's project with humped backs which resemble turtles. They crawl on the floor, manipulated by a child at the terminal. The child draws geometrical figures by manipulating the turtle. As the figures the children wish the turtles to make become more complex, the children receive instructions which go something like this: If you can't solve a problem as it stands, try simplifying it; if you cannot find a complete solution, find a partial one. No doubt, everyone else gives similar advice. The difference here is that the advice is concrete enough to be followed by children who seem quite impervious to the usual mathematics.

Many of you are no doubt familiar with the work on Turtle Geometry. What I find particularly interesting is that Papert and his co-workers also try to teach such activities as walking on stilts, riding a unicycle, and juggling. They do this by constructing people procedures analogous to the computer procedures we've been discussing.

Next I want to discuss a technological advance which will have major implications for education. I'm referring to the widespread introduction of pocket calculators. In the United States the price of such calculators has dropped to a few dollars. This means school children can afford to have one, just as at one time all engineering students had slide rules. What is the implication of this technological advance for the teaching of arithmetic to young children?

When I went to school we spent a significant amount of time learning multiplication tables and the manipulative skills of arithmetic. We learned to do multiplication, division, etc., by rote with no understanding of the algorithm.

I believe that too much time is now spent teaching manipulation of numbers. Arithmetic is not an intellectual activity; for most children it is drudgery. With calculators they have a chance to experiment with and enjoy numbers. Then it might be an intellectual activity to learn the algorithms of arithmetic and for more advanced students, the analysis of these algorithms. It might also be useful for students to learn to do approximate arithmetic calculations or to be able to estimate the order of magnitude of a result. However this does not seem to be taught.

But pocket calculators that do arithmetic and statistical calculations, and that evaluate elementary functions are only the beginning. What will the pocket calculators and home computers of the future be able to do? And what will they imply for education in the future? I suggest we should be thinking hard about that now.

I said earlier that we should not be teaching arithmetic. What should we be teaching? The <u>New Mathematics</u> focused on teaching children set theory. It seems to me that if one wanted to create a New New Math it might be based

on the algorithmic and heuristic point of view. It would teach paradigms for problem solving, both in general, and for specific domains.

My own experience is that I learn algorithmically. Furthermore I find that students learn best when they are taught paradigms. Finally, I feel that the algorithmic viewpoint is so general that I can understand at least some of the issues in a broad array of disciplines. D. Knuth says it very well (21):

"A person well-trained in computer science knows how to deal with algorithms: how to construct them, manipulate them, understand them, analyze them. This knowledge prepares him for much more than writing good computer programs; it is a general-purpose mental tool which will be a definite aid to his understanding of other subjects, whether they be chemistry, linguistics, or music, etc. The reason for this may be understood in the following way: It has often been said that a person does not really understand something until he teaches it to someone else. Actually a person does not *really* understand something until he can teach it to a *computer*, i.e., express it as an algorithm.

. . .

For three years I taught a sophomore course in abstract algebra, for mathematics majors at Caltech, and the most difficult topic was always the study of "Jordan canonical form" for matrices. The third year I tried a new approach, by looking at the subject algorithmically, and suddenly it became quite clear. The same thing happened with the discussion of finite groups defined by generators and relations; and in another course, with the reduction theory of binary quadratic forms. By presenting the subject in terms of algorithms, the purpose and meaning of the mathematical theorems became transparent.

. . .

These examples and many more have convinced me of the pedagogic value of

an algorithmic approach; it aids in the understanding of concepts of all kinds. I believe that a student who is properly trained in computer science is learning something which will implicitly help him cope with many other subjects."

Concluding Thoughts

I have presented examples of the use of algorithmic and heuristic procedures in mathematics, science and applied science, and education. These procedures range from completely analyzed algorithms to heuristic procedures which work most of the time.

We are interested in how to use knowledge to solve problems. Reddy (22) distinguishes among four types of knowledge: algorithmic, formal, informal, and new knowledge. Problems for which we have well defined step by step operations such as the FFT or a business payroll procedure are examples of algorithmic knowledge. Formal knowledge in text books is routinely taught and applied but is not readily expressed in algorithmic form. Such knowledge can, nevertheless, be used by machines in which knowledge rules are activated when their preconditions are satisfied. All of us routinely use a great deal of informal knowledge which is not taught but learned from observation and example. Much of the human sensory and motor activity is of this form. Use of knowledge to create new knowledge (as in all research) is perhaps the most challenging form of problem solving activity. Lenat's work on discovery is an example of this type activity. One of the strong influences of computers on society will be the development of procedural and constructive forms of formal and informal knowledge.

Although the procedural point of view was forced on us by the computer, I believe it will turn out to be a very fruitful direction for us. A naive view of procedure is that it's rigid. In fact, the domain of procedures is very rich and as computer and languages become more complex we will be able

to achieve flexibility and creativity of a degree that would once have seemed mysterious.

What view of the world is the procedure oriented view replacing? Partially, it is replacing a mystical view of creativity. It may be possible for us to use our creativity and imagination in the coming years to really understand the heretofore elusive nature of human creativity and imagination.

Acknowledgment

I wish to thank D. R. Reddy and M. Shamos for stimulating discussions on the subject of this paper and E. A. Feigenbaum and J. Wavrik for comments on the manuscript.

References and Notes

1. D. E. Knuth, *The Art of Computer Programming, Volume 1, Fundamental Algorithms* (Addison-Wesley Publishing Company, Reading, Mass., 1968) pp. 1-9.
2. E. A. Feigenbaum and J. Feldman, *Computers and Thought* (Mc-Graw-Hill Book Company, Inc., New York, 1963), pp. 6-7.
3. M. O. Rabin, *Comm. Assoc. Comput. Mach.* 20, *625 (1977).* [This paper was Rabin's Turing Lecture.]
4. J. F. Traub (ed.), *Algorithms and Complexity: New Directions and Recent Results* (Academic Press, New York, 1976).
5. M. J. Fischer and M. S. Paterson, in *Complexity of Computation*, R. M. Karp, ed. (Amer. Math. Soc., Providence, R.I., 1974), pp. 27-41.
6. Panel discussion in *Perspectives on Computer Science*, A. K. Jones, ed. (Academic Press, New York, 1977), pp. 241-243.
7. H. R. Lewis and C. H. Papadimitriou, *Sci. Amer.* 238, 96 (January 1978).
8. G. B. Kolata, *Science* 197, No. 4305, 747 (1977).
9. R. P. Brent, in *Analytic Computational Complexity*, J. F. Traub, ed. (Academic Press, New York 1976), pp. 151-176.
10. R. P. Brent and J. F. Traub, *On the Complexity of Composition and Generalized Composition of Power Series*, Tech. Rept., Comp. Sci. Dept., Carnegie-Mellon University, Pittsburgh, Pa., May 1978.
11. H. T. Kung and J. F. Traub, *J. Assoc. Comput. Mach.* 25, 245 (1978).
12. K. I. Appel and K. Haken, *Bull. Am. Math. Soc.* 82, 71 (1976). See also T. L. Saaty and P. C. Kainen, The Four-Color Problem, Assaults and Conquest (McGraw-Hill Book Company, Inc., New York, 1977).
13. D. B. Lenat, in *Proceedings of IJCAI, Volume Two* (1977), pp. 833-842.
14. E. A. Feigenbaum, in *Proceedings of the 1978 National Computer Conference* (AFIPS Press, Montvale, N.J., 1978), pp. 227-240.

15. B. G. Buchanan, D. H. Smith, W. C. White, R. J. Gritter, E. A. Feigenbaum, J. Lederberg, and C. Djerassi in *Journal of the* ACS 98, 6168 (1976).

16. R. Risch, *Trans. Amer. Math. Soc.* 139, 167 (1969). See also J. Moses, *Comm. Assoc. Comput. Mach.* 14, 548 (1971).

17. E. Shortliffe, *Computer-based Medical Consultations: MYCIN* (Elsevier, New York 1976).

18. R. O. Duda, P. E. Hart, N. J. Nilsson, R. Reboh, J. Slocum, G. L. Sutherland, *Development of a Computer Based Consultant for Mineral Exploration,* SRI International, Projects 5821 and 6415.

19. W. Clancy, in *International Journal of Man-Machine Studies*, January 1979 (forthcoming).

20. P. A. McCorduck, *Machines Who Think* (W. H. Freeman, San Francisco, 1979).

21. D. E. Knuth, *Amer. Math. Mo.* 81, 323 (1974).

22. D. R. Reddy. Private communication.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) CMU-CS-79-102 / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) THE INFLUENCE OF ALGORITHMS AND HEURISTICS	5. TYPE OF REPORT & PERIOD COVERED (9) Interim rept.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) J.F. Traub	8. CONTRACT OR GRANT NUMBER(s) (15) N00014-76-C-0370 - NSF-MCS75-222-55
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Dept. Pittsburgh, PA 15213	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217	12. REPORT DATE (11) January 1979
	13. NUMBER OF PAGES 18 (12) 18p.
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same as above	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

403 081

JOB